Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №5

Технології розроблення програмного забезпечення

Тема: Патерни проектування

Виконав
студент групи ІА-34:
Жительний В.В.

Перевірив:
Мягкий М.Ю.

Київ 2025

Мета: Вивчити структуру шаблонів «Adapter», «Builder», «Command», «Chain of responsibility», «Prototype» та навчитися застосовувати їх в реалізації програмної системи.

Тема:

21. **Online radio station** (iterator, adapter, factory method, facade, visitor, client-server)

Додаток повинен служити сервером для радіостанції з можливістю мовлення на радіостанцію (64, 92, 128, 196, 224 kb/s) в потоковому режимі; вести облік підключених користувачів і статистику відвідувань і прослуховувань; налаштувати папки з піснями і можливість вести списки програвання або playlists (не відтворювати всі пісні).

Хід роботи

Для забезпечення гнучкості системи у питаннях інтеграції із зовнішніми споживачами даних було обрано структурний патерн проєктування "Адаптер". Даний шаблон доцільно використовувати, коли система має внутрішнє представлення даних, яке є несумісним з інтерфейсом або форматом, що його очікує зовнішній клієнт. Його застосування дозволяє трансформувати дані у необхідний вигляд без зміни внутрішньої логіки бізнес-компонентів.

У розробленій системі цей шаблон застосовано для уніфікації механізму експорту аналітичних звітів та реалізовано через наступні компоненти:

- IReportExporter: Цільовий інтерфейс, що визначає контракт взаємодії. Він декларує єдиний метод export(), який очікує отримати дані про станцію та повернути сформований звіт у вигляді рядка. Це дозволяє клієнтському коду (RadioFacade) працювати з будь-яким форматом експорту уніфіковано, не знаючи деталей реалізації.
- JsonReportAdapter: Конкретний адаптер, що реалізує перетворення внутрішньої статистики у формат JSON. Цей формат є стандартом для веб-API та легко обробляється JavaScript-клієнтами.
- XmlReportAdapter: Альтернативний адаптер, що забезпечує експорт тих самих даних у формат XML, який часто використовується для інтеграції з корпоративними системами звітності.

Графічне представлення структури реалізації даного шаблону наведено на діаграмі класів:

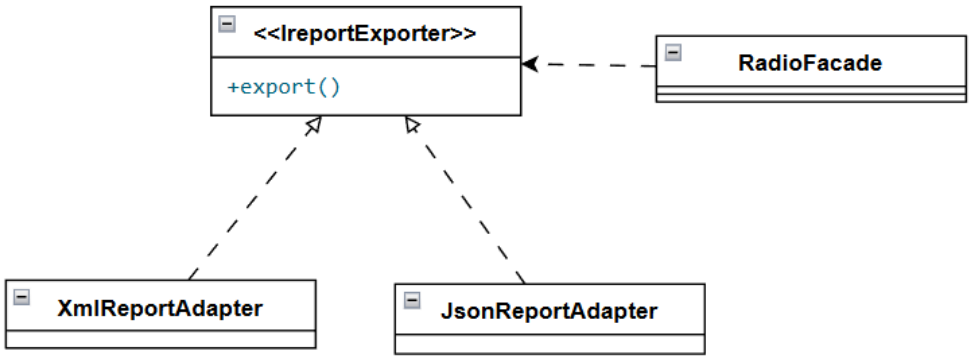


Рис. 1 Діаграма класів для шаблону Ітератор

Коротко описуючи діаграму: Клієнтським класом, що ініціює експорт даних, є RadioFacade. Він взаємодіє виключно з абстракцією IReportExporter, не залежачи від конкретної реалізації формату. Класи JsonReportAdapter та XmlReportAdapter імплементують цей інтерфейс, інкапсулюючи логіку перетворення даних. Така архітектура дозволяє легко додавати нові формати звітів шляхом створення нових класів-адаптерів, не змінюючи код основного фасаду системи.

Програмна реалізація

```
public interface IReportExporter {
    String export(Station station);
}
```

Інтерфейс IReportExporter визначає контракт для підсистеми експорту даних. Він декларує єдиний метод export, який приймає об'єкт станції та повертає сформований звіт у вигляді рядка. Використання цього інтерфейсу дозволяє клієнтському коду абстрагуватися від конкретного формату даних.

```
@Primary

@Component("jsonExporter")

public class JsonReportAdapter implements IReportExporter {


    @Override

    public String export(Station station) {

        ContentStatistics visitor = new ContentStatistics();

        station.accept(visitor);
```

```
        String name = visitor.getStationName();

        int tracks = visitor.getTotalTracks();

        int playlists = visitor.getPlaylistCount();

        long duration = visitor.getTotalDurationSeconds();



        return String.format(

                "{\n" +

                        "  \"station\": \"%s\",\n" +

                        "  \"statistics\": {\n" +

                        "    \"total_tracks\": %d,\n" +

                        "    \"total_playlists\": %d,\n" +

                        "    \"duration_seconds\": %d\n" +

                        "  },\n" +

                        "  \"status\": \"active\"\n" +

                        "}",

                name, tracks, playlists, duration

        );

    }

}
```

```
@Component("xmlExporter")
public class XmlReportAdapter implements IReportExporter{
    @Override
    public String export(Station station) {
        ContentStatistics visitor = new ContentStatistics();
        station.accept(visitor);

        return String.format(
                "<StationReport>\n" +
                        "  <Name>%s</Name>\n" +
                        "  <Statistics>\n" +
                        "    <TotalTracks>%d</TotalTracks>\n" +
                        "    <TotalPlaylists>%d</TotalPlaylists>\n" +
                        "    <DurationSeconds>%d</DurationSeconds>\n" +
                        "  </Statistics>\n" +
                        "  <Status>Active</Status>\n" +
                        "</StationReport>",
                visitor.getStationName(),
                visitor.getTotalTracks(),
                visitor.getPlaylistCount(),
                visitor.getTotalDurationSeconds()
        );
```

```
    }
}
```

Класи JsonReportAdapter та XmlReportAdapter є конкретними реалізаціями патерну Адаптер. Їхнє завдання – трансформувати внутрішні дані системи у стандартизовані формати обміну даними, JSON та XML відповідно.

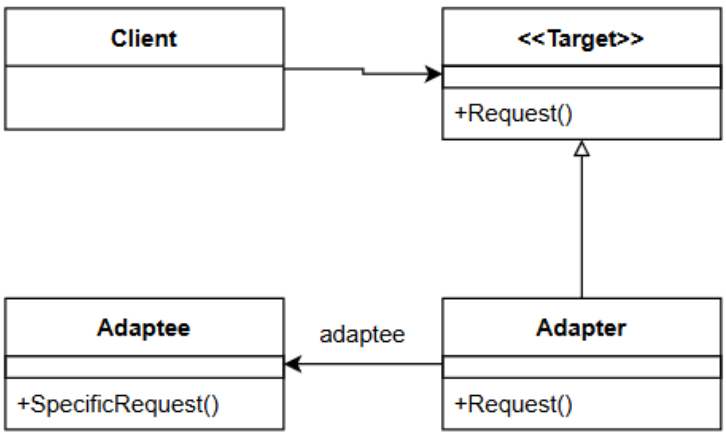
Реалізація обох адаптерів базується на взаємодії з патерном Відвідувач. У методі export створюється екземпляр відвідувача ContentStatistics, який "обходить" структуру станції (плейлисти та треки), збираючи необхідні метрики: загальну кількість треків, кількість плейлистів та сумарну тривалість ефіру. Після завершення обходу адаптер отримує агреговані дані від відвідувача та форматує їх у відповідний текстовий вигляд, повертаючи готовий рядок. Такий підхід забезпечує гнучкість системи, дозволяючи легко додавати нові формати звітів без зміни логіки збору статистики.

Контрольні питання:

1. Яке призначення шаблону «Адаптер»?

Шаблон «Адаптер» використовується для адаптації інтерфейсу одного об'єкту до іншого. Він дозволяє об'єктам з несумісними інтерфейсами працювати разом, приводячи їх до уніфікованого вигляду.

2. Нарисуйте структуру шаблону «Адаптер».



3. Які класи входять в шаблон «Адаптер», та яка між ними взаємодія?

- Client: Працює з об'єктами через загальний інтерфейс.
- Target: Загальний інтерфейс, який очікує клієнт.

- Adapter: Реалізує інтерфейс Target і зберігає посилання на об'єкт Adaptee.
- Adaptee: Специфічний компонент, методи якого потрібно викликати.

Взаємодія: Клієнт викликає метод у Адаптера через інтерфейс. Адаптер перенаправляє цей виклик, викликаючи специфічні методи у компонента Adaptee, який він "обгортає". Клієнт при цьому не знає про деталі реалізації компонента.

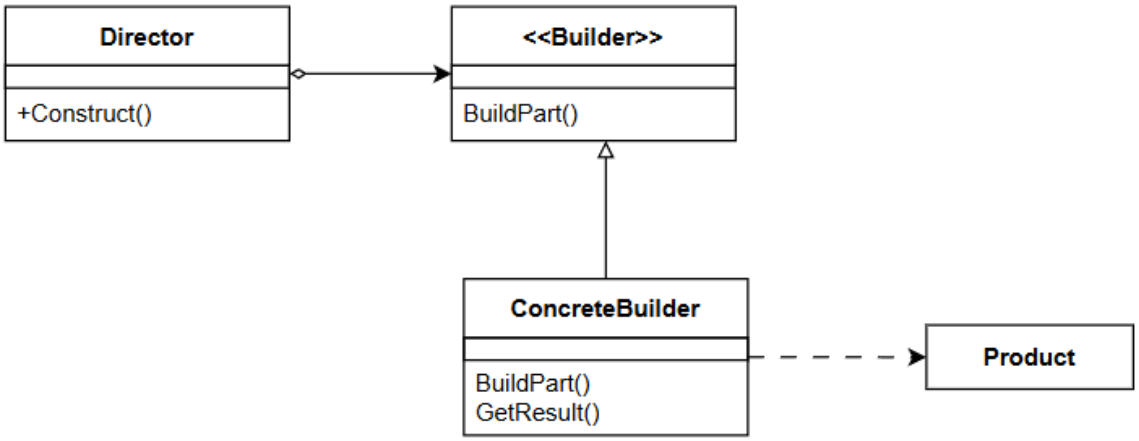
4. Яка різниця між реалізацією «Адаптера» на рівні об’єктів та на рівні класів?

Адаптер на рівні об'єктів працює за принципом агрегації. Адаптер містить посилання на об'єкт компонента, тобто обгортає його і викликає його методи. Адаптер на рівні класів зазвичай реалізується через множинне спадкування, тоді як на рівні об'єктів – через "обгортку".

5. Яке призначення шаблону «Будівельник»?

Шаблон «Builder» використовується для відділення процесу створення складного об'єкту від його представлення. Це дозволяє використовувати один і той самий процес конструювання для створення різних відображень об'єкта.

6. Нарисуйте структуру шаблону «Будівельник».



7. Які класи входять в шаблон «Будівельник», та яка між ними взаємодія?

- Builder: Інтерфейс, що оголошує етапи створення продукту, наприклад методи для додавання заголовків, коду статусу тощо.
- ConcreteBuilder: Реалізує етапи побудови для конкретного типу продукту.
- Director: Керує процесом створення, викликаючи методи будівельника у певному порядку.
- Product: Складний об'єкт, що створюється, наприклад, web-сторінка.

Взаємодія: Алгоритм (web-сервер) звертається до Будівельника, викликаючи методи для побудови окремих частин як заголовки та тіло. Будівельник крок за кроком формує об'єкт.

8. У яких випадках варто застосовувати шаблон «Будівельник»?

Коли об'єкт має складний процес створення наприклад, web-сторінка з багатьох частин.

Коли об'єкт повинен мати декілька різних форм створення наприклад, як конвертація даних з одного формату в інший.

Коли потрібно мати гнучкий контроль над процесом створення продукту.

9. Яке призначення шаблону «Команда»?

Призначення шаблону – перетворити звичайний виклик методу в об'єкт. Це дозволяє параметризувати об'єкти діями, ставити запити в чергу, логувати їх, а також підтримувати операції скасування.

10. Нарисуйте структуру шаблону «Команда».

11. Які класи входять в шаблон «Команда», та яка між ними взаємодія?

- Command: Інтерфейс для виконання операції.
- ConcreteCommand: Реалізує інтерфейс, зв'язує дію з Receiver.
- Invoker: Викликає команду. Не знає конкретного класу команди, працює через інтерфейс.
- Receiver: Об'єкт, який фактично виконує бізнес-логіку.

Взаємодія: Invoker викликає метод виконання у об'єкта Command. Команда перенаправляє виклик до Receiver, який виконує реальну роботу.

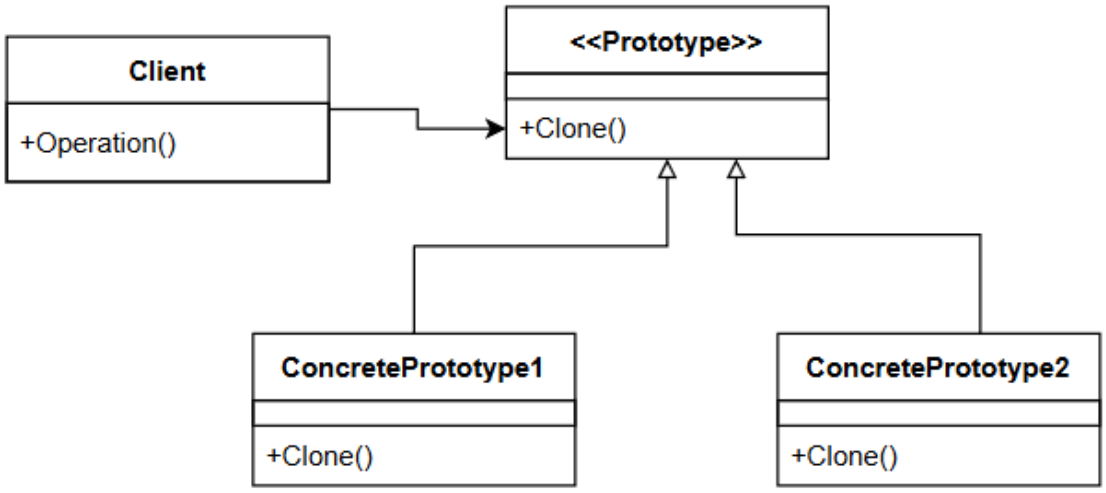
12. Розкажіть як працює шаблон «Команда».

Шаблон інкапсулює запит як об'єкт. Об'єкт команди не виконує дію сам, а перенаправляє запит одержувачеві. Це дозволяє відділити візуальну частину, наприклад кнопки, від логіки обробки. Завдяки цьому можна створювати черги команд, логувати історію дій або реалізовувати скасування дій.

13. Яке призначення шаблону «Прототип»?

Використовується для створення нових об'єктів шляхом копіювання існуючого об'єкта-зразка. Це дозволяє уникнути прямого створення об'єктів через конструктор і зменшити залежність від конкретних класів.

14. Нарисуйте структуру шаблону «Прототип».



15. Які класи входять в шаблон «Прототип», та яка між ними взаємодія?

- Prototype: Інтерфейс, що оголошує метод Clone().
- ConcretePrototype: Реалізує метод клонування, копіюючи свої поля.
- Client: Створює новий об'єкт, звертаючись до прототипу з запитом клонувати себе.

Взаємодія: Клієнт зберігає посилання на базовий тип Prototype, а не конкретний клас. При необхідності створення нового об'єкта клієнт викликає метод Clone(). Прототип створює свою копію і повертає її клієнту.

16. Які можна привести приклади використання шаблону «Ланцюжок відповідальності»?

Формування контекстного меню у складних UI-формах. Клік передається від вкладеного елемента, як приклад Text, до батьківського TextBlock, потім до контейнера Grid, і кожен елемент може додати свої пункти в меню або передати запит далі.

Висновок: Під час виконання даної лабораторної роботи було досягнуто поставлену мету: вивчено та проаналізовано групу шаблонів проєктування, що використовуються для вирішення типових задач в об'єктно-орієнтованому програмуванні. Було детально розглянуто призначення, структуру та принципи взаємодії класів для наступних патернів: «Адаптер», «Будівельник», «Команда», «Ланцюжок відповідальності» та «Прототип». А також на практиці в програмному коді було реалізовано шаблон «Адаптер».